Bachelor thesis

*Independent degree project*

*Computer Engineering*

Evaluation and Implementation for Pushing Automatic Updates to IoT Devices

Menglei Min
Abstract

In recent years, Internet of Things has developed rapidly, and now has penetrated into human life and industrial production. It is speculated that the internet of things will become ubiquitous in the future, which will bring a series of problems. First, the large number of things will lead to operated system and software updates consuming a lot of manpower and resources. Another problem is the Internet of things facing security issues, in recent years for the means of Internet of things and tools have been increasing largely. Therefore, to achieve a secure automatic update on the Internet of Things is essential. This report will follow such an automatic update system based on Internet of things to expand. First it elaborated on the main motive of this problem, found three existing related works and three security methods for communication to analyze. Then combined results of analysis, put forward own a secure automatic update solution: manager and devices connect and mutual authentication in real time, at the same time, the manager will regularly check the database to see if there is new version application. When the administrator uploads a new version, the manager will download the version and then sends to all devices, then device installs and finally restart itself. Next, the report described how to implement this system in detail and evaluated it. In the end, this report summarized and introduces the future work.

Keywords: Automatic update, Internet of things, Digital signature, Secure sockets layer communication, Secure hash algorithm
Acknowledgements

I really would like to thank my supervisor Dr. Stefan Forsström for all his help. Without his help, my progress cannot be so smooth. At first, I knew nothing about my thesis work, and didn’t know what to do. He gave my direction and his advice and then I slowly on the right track. He had been patiently guiding for my project and giving feedback on my report writing, besides, helping me modify PPT. In addition, I would like to appreciate professor Tingting Zhang, provide me the chance to study here and help me to select this thesis topic. Besides this, I would like to express my gratitude to my friend Loring for always working with me and giving me advices. Lastly but none the least, thanks Johnny for always supporting and encouraging me when I was down, and thanks for all those who helped me during this period!
# Table of Contents

Abstract ............................................................................................................. iii  
Acknowledgements ........................................................................................... iv  
Table of Contents ............................................................................................... v  
Terminology ...................................................................................................... vii  

1 Introduction ............................................................................................ 1  
1.1 Background and problem motivation ....................................................... 1  
1.2 Overall aim .............................................................................................. 2  
1.3 Concrete and verifiable goals ................................................................... 2  
1.4 Scope ........................................................................................................ 2  
1.5 Outline ...................................................................................................... 3  

2 Theory ..................................................................................................... 4  
2.1 Internet of things ...................................................................................... 4  
2.2 Secure communication ............................................................................. 6  
2.2.1 Secure Hash Algorithm ........................................................................ 7  
2.2.2 Digital Signature .................................................................................... 7  
2.2.3 Secure Sockets Layer ........................................................................... 10  
2.3 Automatic update ................................................................................... 12  
2.3.1 Dynamic software update ..................................................................... 12  
2.3.2 Automatic software update ................................................................. 13  
2.4 Related work .......................................................................................... 14  
2.4.1 Windows Update .................................................................................... 14  
2.4.2 Java Web Start ........................................................................................ 15  
2.4.3 Getdown ............................................................................................... 15  

3 Methodology .............................................................................................. 17  
3.1 Literature study ...................................................................................... 17  
3.2 Solution design ....................................................................................... 18  
3.3 Solution implementation ........................................................................ 18  
3.4 Evaluation ............................................................................................... 18  

4 Implementation .......................................................................................... 19  
4.1 Administrator ......................................................................................... 20  
4.1.1 Application ............................................................................................. 21  
4.1.2 Administrator ......................................................................................... 21  
4.2 Database ................................................................................................. 22  
4.3 Manager .................................................................................................. 23  
4.3.1 Downloader ........................................................................................... 24  
4.3.2 Sender .................................................................................................. 25  
4.4 Authentication .......................................................................................... 25
Terminology

Abbreviations

ASF  Apache Software Foundation
DSA  Digital Signature Algorithm
DSU  Dynamic software update
GUI  Graphical User Interface
HTTP HyperText Transfer Protocol
IDE  Integrated Development Environment
IOS  IPhone OS
IOT  Internet of Things
ITU  International Telecommunications Union
JAR  Java Archive
JKS  Java KeyStore
JNLP Java network launching protocol
JRE  Java Runtime Environment
JWS  Java Web Start
MD5  Message-Digest Algorithm
MIT  Massachusetts Institute of Technology
OOO  Three Rings Design
RFID Radio-Frequency Identification
SHA  Secure Hash Algorithm
SSL  Secure Sockets Layer
SWT  Standard Widget Toolkit
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP/IP</td>
<td>Transmission Control Protocol/Internet Protocol</td>
</tr>
<tr>
<td>TLS</td>
<td>Transport Layer Security</td>
</tr>
<tr>
<td>UI</td>
<td>User Interface</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform / Universal Resource Locator</td>
</tr>
</tbody>
</table>
1 Introduction

For nearly a decade, Internet of Things has developed rapidly, which also has gradually been widespread concern. It is predicted that by 2020, there will be 50 billion devices on the Internet [1]. Internet of Things connects all the items and the Internet to achieve intelligent identification and management through identification technology and other sensing equipment [2]. So this report will focus on the Internet of Things field and propose an implementation solution about secure automatic update system for Internet of Things devices.

Internet of Things is a platform that makes the device smarter and makes communication more effective. Today, many systems based on Internet of Things are put into used, they are more or less involved in our studying, work and daily life, ubiquitously influences on our life way [1]. For example, the popular wisdom city, wisdom campus and wisdom home project have attracted a large number of programmers to join in recent years, coupled with the continuous improvement of the Raspberry Pi platform and many raspberry-pi-based cloud service platform came into beings. In addition, Internet of Things has also reduced the threshold of knowledge level on the programmers. Above these reasons provide internet of things a broad development prospects.

1.1 Background and problem motivation

In recent years, Internet of Things develops rapidly, which has brought many benefits to mankind. However there are also many problems related to secure communication in the Internet of Things, which will be improved to avoid such problems in the future. In 2016 year, Internet of Things has suffered a lot of attacks, including denial of service attacks through cameras, shared private file routers and leaked password smart applications [3]. On the other hand, we assume that things connects to the Internet of Things is ubiquitous in the future, so the World of things is bound to be huge and complex. The complex update operation of devices is impossible by ordinary users due to the huge workload and expensive time cost. In this case, automatic update is a better choice. This technology can attract the attention of many vendors in the future if Internet of Things develops smoothly, and there will be an increasing demand on the Internet of Things market.
1.2 Overall aim

At present, the rapid developments of the Internet of Things industry and the substantial increasing in devices have brought new challenges to the manual update. In addition, the number of cyber-attacks and threats aiming at the internet of things is more and more, and criticality of internet increased all the time. So the design and implementation solution for internet of things must consider these factors.

The problem of this project is to find and investigate means for secure pushing automatic updates to devices in an Internet of Things system, then propose own feasible implemented solution in a proof-of-concept scenario and measure its performance and security.

1.3 Concrete and verifiable goals

It is a better choice to specify the overall goal into following several sub-goals according to the project progress such that readers can understand it more clearly.

Goal 1: Find three possible solutions for pushing automatic updates to devices.

Goal 2: Find three possible solutions for creating secure IoT communication systems among administrator, server, devices manager, devices.

Goal 3: Identify the most suitable combination of these solutions.

Goal 4: Implement a proof-of-concept system that uses the proposed solution.

Goal 5: Evaluate its performance in terms of its performance such as total latency, intermediate latency, and security according to the installation success rate in different conditions.

1.4 Scope

This project focus on implementing secure automatic update system in the scenario concept instead of in the actual network environment, so some negative effects (e.g. too large number of devices, network overload…) are ignored. In addition, the project just uses relevant tools to simulate some equipment like the sensors and actuators, instead of the real ones, so some unexpected situations (e.g. server crash) were not considered. The scope of evaluation is mainly about the total latency from administrator upload the application to device restarts successfully,
download latency, verification latency, transmission latency, installation latency, restart latency and installation success rate instead of the scalability, availability and so on.

1.5 Outline
Chapter 2 describes the theory knowledge, which is mainly about some standards, reference or some other identical knowledge of automatic update and secure communication. Chapter 3 illustrates the concrete steps for implementing the secure automatic update in IoT system. Chapter 4 is mainly about the proposed solution and explains how to implement the design model in more detail. Chapter 5 presents the results of the secure pushing automatic update system. Chapter 6 evaluates the system according to its performance and security, and analyzes these measured results. Chapter 7 gives some discussion and overall conclusion based on above results. In addition, mentions the ethical problem and prospects some future work related to the project.
2 Theory

This chapter illustrates the detailed theory knowledge and basic background information, which can help reader to better understand the project. The first section will introduce the Internet of things. The second section is about the secure communication, which elaborates some encrypted algorithms and encrypted techniques. The next section is the main part “Automatic update”, which will introduces its basic framework and related approaches and techniques used in implementation part. The final section is the related work, separately explaining the basic theory for three existing works.

2.1 Internet of things

We witnessed the rise of IoT (Internet of Things) era. As its name implying and in short, the Internet of things refers to such a network where everyday objects connect to the Internet each other[4]. IoT has made the internet ubiquitous in human daily life, and this ubiquity of IoT has not only brought convenience to human but also contributed to the development of many areas, such as the chip industry, wireless communication, security, distributed system, cloud computing and so on. The meaning of Internet of things is that it achieves a seamless connection between physical space and digital space. To some extent, the applications of IoT have brought a revolution to the human life. As shown in figure 2.1, the Internet of things benefits human in lots of aspects.

![Figure 2.1: The overview of Internet of Things.](image)
The concept of the Internet of things was first formally proposed by Kevin Ashton, David Brock and Sanjay Sharma in Auto-ID Laboratory at Massachusetts Institute of Technology (MIT) in 1999 [5], the Auto-ID Lab is the worldwide research organization for the field of Internet of things. Then the topic of Internet of things was reported again by the ITU (International Telecommunications Union), and this time the internet of things was hit to another level. After that, the research of internet of things has risen sharply, especially in recent years, a variety intelligent applications are appeared all at once.

The term “Internet of things” consists of two terms, one is the “Internet”, another is the “Things” [2]. There are two meanings of Internet of things according to the two terms. First, the core and foundation part of IOT is still the “Internet”. Second, the users are extended to any “Things” which can get on information exchange and communication. There are many definitions for Internet of things. Strictly speaking, it is defined as: connects items with the Internet for resource exchange and communication according to some specific protocols through information sensing equipment (such as Radio-Frequency Identification), and to achieve some specific purpose [6].

The term “Radio-frequency identification (RFID)” was proposed by Kevin Ashton who is the founders of the Auto-ID Lab [7]. RFID reads information stored in a tag through radio wave. The advantage of RFID device comparing to bar code is that it does require a direct line of sight for tracking reader [8]. The RFID system is composed of two parts. One is a tag, another is a reader illustrated in figure 2.2. Tag is used to collect information, and reader is used to transmit information to computer. After these work done, computer will process information and then execute some specific behavior.
As shown in figure 2.3, the RFID tag refers to a device equipped with antenna and microchip, and uses the RFID tag to track objects [9]. The tag part also has two subparts: one antenna for receiving and transmitting signal, one microchip as the storage for storing and processing information. The tag is specified with unique numbers for identification.

Although the path of internet of things goes fast dramatically, there are still lots of challenges to impede its development. For example, the security issue, power, complexity of IoT framework, connectivity, scalability and so on. So there are lots of things need to do for overcoming these challenges in the future.

2.2 Secure communication

Secure communication refers to two entities which communicate with each other in a secure manner. The longer communication will
introduce higher probability being compromised. For this reason, lots of secure theory and technology about encryption are proposed. This section will introduce several mainstream encryption algorithms which will be applied in this project.

2.2.1 Secure Hash Algorithm

Secure Hash Algorithm is abbreviated as SHA, which is one of cryptographic hash functions family [10]. It can be applied for Digital Signature Algorithm (DSA). Actually, the message digest is a fixed length of string called “Hash Value”. For example, for a message with less than $2^{64}$ bits length, SHA1 as a family of SHA will generate a 160 bits message digest as figure 2.4.

![Figure 2.4: The message digest basic generation process.](image)

The message digest is very important as it will be used to verify the data integrity when the message is received on receiver. The message digest is stored in specific place of message or file and is transmitted with the carrier such as message or file. If the message is corrupted, then the receiver will produce a different message digest when re-compute it.

SHA has some good characteristics:

**Irreversibility:** The original message cannot be restored from the message digest. So far, SHA is still considered to be quite safe unlike MD5 Message-Digest Algorithm which has been cracked.

**Uniqueness:** Two different messages will never produce a same message digest. This is a big probability hypothesis, because there is still extremely little chance to appear the same message digest, but generally it is ignored.

2.2.2 Digital Signature

The signature in the traditional sense refers to a handwriting that describes someone’s name or else mark, which are usually written on the documents as a proof of someone’s identity. Likewise, digital signatures also play such an important role as traditional signature
except their different manifestations. Digital signature is used to demonstrate the digital document’s authenticity. A valid digital signature should guarantee three secure elements of message: Integrity, Confidentiality, Non-repudiation.

**Integrity:** No unauthorized modification of information. In other words, guarantee the source, destination of the message is correct and the message content cannot be tampered.

**Confidentiality:** No unauthorized disclosure of information. In other words, guarantee that the message will not be leaked by unauthorized entity.

**Non-repudiation:** Guarantee that the sender and receiver of message cannot deny that they have done the operation.

Digital signature is one of the applications of asymmetric encryption algorithms, also known as public key cryptography. A digital signature technology usually refers to three algorithms, key generation algorithm, signing algorithm and verifying algorithm, which are expressed by \( (G, S, V) \) \[11\].

**Key generation algorithm.** Key-generator \( G \), using a public key algorithm, such as the famous RSA to generate a pairs of keys: the private key prk and corresponding public key pk.

**Signing algorithm.** Signing \( S \) signs a message with the private key to produce a signature. Input string s and private key prk, returns signature ss.

**Verifying algorithm.** Verifying \( V \) verifies the signature with public key, and then claims the authenticity of the message according to different results. Input string s, signature ss and public key pk, and output accept or reject.

The digital signature signing stage is illustrated as figure 2.5. After generating the public key and private key, the signing tool will create a one way hash value of original data with hash function. Then encrypt the hash value using private key to produce an encrypted hash which is called signature. The certificate will be used to decrypt the signature in the verifying stage.
The digital signature verifying stage is illustrated as figure 2.6. The hash value is unique and irreversible. This means that any data change even a single character will cause a huge change. This feature helps receiver to validate the data integrity by decrypting the digital signature with sender’s public key which is the certificate and get hash value $h$. As the same time, the hash value of original data which is fetched from signed data is computed again, assuming the hash value is $h'$. If $h$ equals to hash value $h'$, then judge that the data is not changed. If $h$ doesn’t equal to hash value $h'$, then the data is seen as tampered with by someone, so the data integrity is destroyed.

In addition, it also can prove whether the private key matches the corresponding public key, so the authentication can be guaranteed. Besides, it is very difficult to deny having signed document because the owner of specific private key is unique, so the Non-repudiation is guaranteed.
The application of digital signature is extremely extensive. Nowadays, in many countries, digital signatures of electronic documents have been seen as the legal signatures, and relevant United States departments have published some laws and regulations of digital signature.

2.2.3 Secure Sockets Layer

Communication system is a crucial part to the internet, and how to ensure communications security is always an important issue. Along with the establishment of modern communication system, people found some feasible and efficient ways to guarantee digital communication security using some mathematical theory. The basic concept is that encrypt the communication content between two sides in case the content will be easily intercepted by eavesdroppers.

Secure Sockets Layer abbreviated as SSL is one of the ways to solve above problems. SSL is a kind of security protocol, which is designed to provide security for internet communications. It was first proposed in 1994 by Netscape and then is widely used in digital communication. At present, SSL and subsequent version TLS (Transport Layer Security) are relatively mature secure communication protocol and has been designed to interface to facilitate the use of programmers in multiple programming language including Java. They often serve as client-server mode to establish an encrypted communication channel.

From the perspective of functional aspects, SSL/TLS protocol can be departed into two sub-protocols: the SSL/TLS Record Protocol and Handshake Protocol [12] as figure 2.7. The Record protocol is based on the TCP/IP protocol, which provides reliable data transport channel and some other basic function. The Handshake protocol rely its lower Record protocol, which is used to authenticate with each other, negotiate related algorithm and exchange session keys before actual data transmission starting.

![Figure 2.7: Two sub-protocols of secure socket protocol.](image-url)
From the point view of certification direction, the SSL/TLS communication can also be divided into two types: SSL/TLS one-way authentication and two-way authentication [13]. Figure 2.8 illustrates these two types of authentication way.

**SSL/TLS one-way authentication:** There is only unilateral certification between two sides. First, the client initiates a handshake request. Second, after server receiving it, server will choose suitable protocol version and encrypted method for this specific client. Then the server will send the negotiation result and its certificate to this client. Third, the client will send session key encrypted with server’s public key (certificate) to server. Fourth, server will decrypt encrypted data with its own private key. Now, these two sides can start a secure SSL communication with the session key which is only known by these two sides, so subsequent transmitted data will be encrypted with session key.

**SSL/TLS two-way authentication:** Both sides will be mutual authentication. In other words, both of two sides will exchange public keys (certificates). This basic process is likely to one-way SSL/TLS authentication, only little difference is several additional steps in negotiation phase as shown in the dashed box in figure 2.8. After the server sending negotiation result and certificate to client, it will request the client’s certificate. And then client sends its certificate to server. In addition, the client will send its digital signature generated with its own private key to the server, and server will verify the legitimacy of digital signature using client’s certificate (public key). Subsequent steps are same as one-way authentication. Finally, these two sides establish the secure socket communication and can start exchange message in a secure manner.
2.3 Automatic update

Now many operating systems and applications provide the automatic update function to give better user experience or some other reasons. Some necessary basic knowledge related to automatic update needs to be grasped.

2.3.1 Dynamic software update

Dynamic software update is abbreviated as DSU. DSU refers to the research field that the updating program must be executed when it is running. In the past, there is not much research and application on DSU because the number of devices is not large even small. However now, things are different, the internet of things will bring huge number of devices, So DSU shows a broader space as the internet of things develops.

The goal of DSU is to make update process to be done without interrupt. The Dynamic update process can be considered as including three parts according to the reference [14] which are dynamic linking, re-linking, and state transfer respectively.

**Dynamic linking:** The new version resource should be made accessible to the specific program when it is running. More specifically, a tool which is designed to load data firstly accesses resource from the shared libraries, and then put these into specific place such as the address space. After that, the program can locate these resources from library only by their names, because the subsequent work is handed over to specific tool or mechanism.
Re-linking: After the dynamic linking process to be done, there will isn’t a loading problem for the new resources such as the code, because the program will be linked to correct place. But for the old resources, it is not same, so Re-linking process is needed. The Re-linking process will make the existing blinder in the old version to re-link to the new place.

State transfer: Generally speaking, the data structure is most likely need to be changed after a program update process done. For example, one type of data structure is integer form in the old version, but which is changed to the string form in the new version. Another example is storage expansion. The program database maybe becomes huger after updating to new version.

2.3.2 Automatic software update
To some extent, automatic software update is an extension of dynamic software updates, but the difference is that automatic update needs to terminate the current version program when the update process is performed, while the dynamic update does not need to do this. Principles of most automatic update applications or systems are similar. Its basic working principle can be divided into two parts according to main function, the downloader and updater.

Downloader: Firstly, the downloader will check for the new version application which usually appears as a jar file on the database regularly. The database may be the web site (URL) or some other form storage. Secondly, the downloader will verify the file for security issue. The form of validation will vary according to the different application or system requirements. Thirdly, after passing the validation, the downloader will download the new version application file, and store it in specific place. What is more, it will notify the updater to update it.

Updater: Firstly, the updater will have been waiting for the downloader’s notice. If it gets the notification from downloader, then it will check the timeliness of new downloaded application. If the current version number is less than that of the downloaded application, it will proceed to the next step. Secondly, the updater program will detect whether the current version program is running. If it is running, the updater will terminate the running program and do a backup for current version application in case the update process fails. Thirdly, the updater will execute the install process of the new version application. During this period, it can output the some specific information
according to the user requirement. Even display nothing so that makes user feel transparent. Fourthly, the updater will restart the device when the install process to be done, and makes device run on the new version application when device startup.

In short, the downloader is responsible for downloading the new version application or system, and the updater is responsible for updating it.

2.4 Related work

In order to better propose solution and implement this project, we need to study several existing related works. These existing successful works could be used as a part of solution to this project’s problem statement. We will first study the Windows Update solution, followed by Firefox Update solution, finally Java Web Start solution.

2.4.1 Windows Update

Traditional operating systems typically contain some kind of automatic update solution. For example, in daily life, we may often encounter on system updates, such as from Windows 7 to Windows 10 or IOS 10.0 to IOS 10.3.1 Windows Update is one of the most widely known auto-update applications, but at first Windows Update does not support automatic updates until the release of ME in 2000.

The Windows Update includes automatic downloading and installation. The auto-update client will check update items to the Windows Update server every day. Then the user can choose to download the available update items and immediately install or just notify user to download these items in advance.

More specifically, when user turns on the Windows Update function, then it will connect the user’s computer to the website server. Some specific tools will scan user’s operating systems and check whether there exists some items need to be upgraded or not. If exists, then it will prompt to user to download these update file. When downloading these files and installation finished, this is not the end, the user must restart its computer to upgrade the environment so that complete this time update. For install update, user needs to click a button which indicates the install function and then install it. But actually, the Windows Update file cannot be executed because these files names are ended with .MSU not .exe format [15]. This kind of suffix is designed for Windows Update.
2.4.2 Java Web Start

Java Web Start abbreviated as JWS which is a software technology, in other words it is an application. It is easy to get Java Web Start as long as user’s computer installs Java 5.0 or upper version, because the JWS is plug-in Java Runtime Environment (JRE). This means that Java Web Start will be auto-installed in computer when Java is installed.

Java Web Start makes user download related application files from Website and launches the application in user local system. Java Web Start application has several advantages. First, it provides a “Click Once” convenient way to activate and run application. Second, JWS can guarantee the system is always to run the newest version application. Third, it is easily to be used, avoid the complex install and update process [16]. There are three different ways to launch the application: from the Web browser, desktop and Java Cache Viewer. This report will focus on the Web browser launch approach.

The Java Web Start requires developer setting a JNLP (Java network launching protocol) file, this file is considered as a trigger [17]. JNLP file will trigger the download process for specific application which is often appears as Jar or Ear files. Also, the JNLP file can be seen as the bridge between Java Web Start and Web browser (server), which guides the Java Web Start to where download application files. For secure issue, these files must be signed otherwise the Java Web Start will refuse to download an unsigned files. The Java Web Start application will check regularly whether there is any newer version file. If there is, then it will download these files and replace the older ones which are cached in local system, so these files are always up to date. What is more, a tool named jardiff supports developer to create different and increment version application, based on this technique, it can guarantee the Java Web Start application gets only one version application at one time.

Although the Java Web Start application is a good tool for automatically update, there are some limitations. One is that this requires a Web server such as Tomcat server to distribute these files. Another limit is that it requires the internet connection which reduces its availability sometime and so on.

2.4.3 Getdown

Getdown [18] is an open source application aiming to implement automatic download and installation for a system, which can be downloaded at Github website. This application developed by OOO
(Three Rings Design) which is a company for online game developing. The main goal of Getdown is to distribute and maintain all the files related application. As mentioned in the previous section, the Java Web Start has some limitations, so Getdown is designed to make up these limitations and as a replacement of JWS.

The Getdown application consists of several important control files such as getdown.txt and digest.txt. The getdown file tells system what the downloaded URL and working directory are, which version Jar file it should download and some other basic information about target Jar file. The digest file was designed for security issue. Sometimes, the Internet may be out of work so that the downloaded files are not completed, which are corrupted. It is unpredictable to launch these corrupted files. So to design a digest file is necessary. The digest.txt contains the digest of each file, which can be used to verify legitimacy of target files.

![Figure 2.9: How Getdown application works.](image)

The Getdown working principle is illustrated in figure 2.9. The Getdown process will firstly read the content of control files to decide some necessary parameters. Then check available files and download these. When download done, validate integrity of each target file based on the result of comparing with digest file. If there is any file corrupted, attempt to delete downloaded files and re-download all files until the validation result is good. Subsequently, if the version of the target Jar file is greater than current version specified in the control file, then proceed to upgrade process. Otherwise, continue to validation process. If process goes smoothly, after update installation, then restart to launch the new version application.
3  Methodology

The overall time plan of this project is that the related materials (About Internet-of-Things, automatic update, secure communication and so on) will be gathered and read in the first week after diploma work starting. Then propose a proof-of-concept scenario and implementation solution about this scenario in the next two weeks. The implementation solution will be evaluated for feasibility in order to further advance the diploma work. Next go to the code programming phase, which will cost about three weeks. Subsequently the code debugging will consume about one week. When all above these done, the report writing can be started, which maybe need about one month. Finally it is the presentation prepare phase which needs about one week.

3.1 Literature study

Find three possible solutions for pushing automatic updates to devices. It is necessary to learn knowledge about automatic update and understand the relevant theory, especially try to find some existing automatic update system products. For example, search some relevant papers through Google and the Google scholar, and find some relevant applications in specific platform such as GitHub, stackoverflow and so on. Then to learn what is automatically update and how it works based on these existing automatic products. Finally, summarize the merits of each paper and application, which can pave the way for proposing implement solution.

Find three possible solutions for creating secure IoT communication systems among administrator, server, devices manager, devices. It is also important to learn security knowledge of the Internet of Things system [19], especially for the commonly used encryption algorithms and techniques, such as Secure Hash Algorithm (SHA), Secure Sockets Layer (SSL) communication and Digital Signature Algorithm (DSA). Try to find some related papers and some existing products guaranteeing secure communication between nodes which may be administrator, manager and devices. For example, retrieving some related papers by Google and Google scholar and look for some practical applications guaranteeing secure communication. Then get their merits by analyzing, comparing and summarizing these papers and applications.
3.2 Solution design
Identify the most suitable combination of these solutions.

On the basis of above goal 1 and goal 2, combine their advantages of these solutions, and then identify the most suitable combination of these solutions in terms of each solution’s realization degree, implementation technology difficulty, universality and so on. After these work done, design a basic theoretical model according to the solution combination, which illustrates the workflow among administrator, server, device manager and devices.

3.3 Solution implementation
Implement a proof-of-concept system that uses the proposed solution.

Propose a proof-of-concept scenario, and to implement the secure automatic update system based on this scenario. It is worth mentioning that this scenario particularly needs to reflect the high scale and high speed feature of the sensor. Select the appropriate implementation methods and techniques to develop this system. For example, in this project, Maven Shade Plugin will be used to create a single JAR file. The administrator is responsible for signing application and uploading the new version application to specific database, the database will be a website server (Tomcat server). What is more, Java program will be used for simulating concept scenario such as the Administrator, Device and Manager. Eclipse can provide Integrated Development Environment (IDE) for almost all languages and architectures, so choose Eclipse as developing tool will.

3.4 Evaluation
Evaluate its performance and assess security level of this system.

First of all, measure this system according to the calculated total latency and intermediate latency, and analyze these results to see if the system is in good performance. At the same time, assess the security of this system according to whether the measured results are consistent with the expected results, such as the installation success rate in different conditions. And then analyze these results, giving an assessment of security level. Finally summarizes the advantages and disadvantages of the system.
4 Implementation

This chapter firstly overviews the overall structure of this automatic update system in figure 4.1. Then it illustrates the more specific workflow layer by layer in figure 4.2. After above work, it describes the detail implementation method for each layer of this automatic update system according to the workflow structure of figure 4.2. The proposed overall network structure for the automatic update scheme is shown in figure 4.1.

This is assumed scenario occurred in industrial Internet of things. The network contains the administrator, database, internet, and manager and many devices. The manager is usually played by the gateway of local network and the devices are often the machine or control devices such as the Raspberry Pi. When the factory needs to update some bug, it just uploads the new application to database, and then all devices will be updated automatically in the end.

![Figure 4.1: Proposed overall network architecture.](image)

This figure 4.2 illustrates the relation among administrator, database, manager and all devices. First, the administrator will interact with database directly. Likely, the manager will interact with the database directly. Then the manager (it is usually the gateway which can provide the internet access to local network devices [20]) will communicate with devices if there is a new version application in database through the Internet. Finally, the device will receive the new version application
from devices manager via local network. Above is the basic relation between two entities.

![Proposed communication workflow](image)

**Figure 4.2: Proposed communication workflow [21].**

The more detailed communication workflow is illustrated in figure 4.2. The chapter 4 follows this figure layer by layer to describe detail implementation. First, the administrator will upload the new version application to the database. At the same time, the manager will regularly check whether the database existing the new version application. If there is a new version application, the manager will authenticate with devices each other. There are two results of authentication. One result is authentication failure, and in this case the current update process of specific device will be terminated. Another is authentication success, and then the manager will send the newer version application file to all local authenticated devices. Then these devices will install with the new version application automatically. When the install process done, the device will be restarted, and finally launch this new version application.

### 4.1 Administrator

Administrator layer is responsible for design and implementation of application (JAR file), and upload JAR file to database. So this section will describe how Application and Administrator are built.
4.1.1 Application

The first thing of implementation is to get the right form application which will be transmitted in communication and automatically updated, and it should be a single JAR. In order to better observe the install information of the application such as progress, version number, the output of the application being installed is shown below figure 4.3.

Figure 4.3: Design of application output.

For convenience and efficiency, this project will choose the Maven Shade Plugin [22] to generate the single JAR file. Maven Shade Plugin provides the capability to package project into “Executable JAR”. And the application output specification is set according to the figure 4.3. The related source code is auto-app in Appendix A.

4.1.2 Administrator

Administrator is a UI (user interface) of uploading application. This project uses the SWT (Standard Widget Toolkit) to develop Administrator’s UI. SWT is an open source tool which is designed to provide developer an efficient way to layout widgets. For example, it can provide button, label, text even layout managers. The UI of administrator consists of two parts: Display Interface and Operation interface.

Display Interface: This is relatively simple and contains only three display elements: Application name, Release time, Current Version.

Control Interface: This process is illustrated in figure 4.4. The upload operation will be triggered by click button “Upload”. Then program will prompt to a selection window for choosing JAR file. If the selected file is
existed in database server, then just prompt to reselect. If not, it will continue to judge that if the selected application version is newer than current version (selected application version number is greater than current version number). If not, like last judgment of existent, it will also prompt to reselect. If it is newer, then the program will sign the JAR file using Jarsigner tool to ensure the integrity of JAR file. If sign success, then execute actual “upload” command. If upload fails, then the upload operation will be cancelled. If succeeds, then the signed JAR file will be sent to server and stored to database. At the same time, the Administrator will log the uploaded application information. The related source code is Administrator in Appendix A.

![Functional Flow Chart of Administrator](image)

**Figure 4.4:** The functional flow chart of Administrator.

### 4.2 Database

This database layer is implemented through Apache Tomcat, also called Tomcat Server. Tomcat Server is developed by Apache Software Foundation (ASF) [23]. The mainly purpose of Tomcat Server is to help user quickly develop a Web Application. The Web Application is not like ordinary standalone application because it runs on the Internet, such as Amazon, Google.

Figure 4.5 shows the basic function design of uploading and downloading file in Tomcat server. These functions are a part of
implementing automatic update system. There are two sides in this design: client side and server side. Client side contains Administrator and Manager. Server side appears in the form of “hostname: port” and has a Database and application. The database consists of two parts: the collection of JAR files and one configuration file which indicating what the current newest version is. The difference and innovation between this project and conventional automatic update system is that client does not need to search all application files one by one, instead of check the version content of configuration file. The application is used to display these newest files information in a visual way.

Firstly, the client side (Administrator) initiates a request which contains uploading Jar file to server. After server receiving it, the server will store the Jar file into its database and response to client to claim that server has received and stored it. As the same time, the application of server will display the newest application information. Secondly, the Manager initiates an URL request towards to configuration file of database to check what the newest version is in database and to decide if it needs to download new Jar file. If so, then Manager will download it from the server.

![Figure 4.5: The basic functions of database server.](image)

### 4.3 Manager

Manager layer is the core of this automatic update system. It connects the database and all devices as a bridge. From point view of function, Manager has two main functions: download JAR file (Downloader) and send it to devices (Sender). This section will describe how manager implement these two parts according to figure 4.6. The related source code is Manager in Appendix A.
4.3.1 Downloader

The downloader is designed to download newest JAR file of database. The function of downloader is illustrated in left side of figure 4.6. The work flow of downloader can be divided into three stages: prepare stage, download stage, verification stage.

Stage 1, the downloader will initiate necessary parameters for the next series of job in prepare stage, such as working directory, URL of web server and so on. Then when start to execute checking process, it will check the version number of configuration file in database every period of time. If repository of downloader has existed the newest version JAR file, then it just ignore it and wait to check of next turn. If not, then it will jump to download stage.

Stage 2, download stage will download the newest JAR specified in configuration file using HTTP protocol. And store this JAR file to its repository.

Stage 3, after the JAR file having been downloaded, the downloader will verify its integrity in case that the file is corrupted but still to be launched. The more detailed explanation of verification process will be
discussed in section 4.4.2. If it is corrupted due to some reasons, then it will delete this JAR file and all related files, and go back to check process of stage 1. If not corrupted, then the downloader will write log file to record the download information of this time, such as download time, application name, and version and so on. In addition, it will refresh the display which always shows current newest version information. As the same time, it will trigger sender to send file to device.

4.3.2 Sender

The sender is used to implement the actual function of sending files. Its function is shown in right side of figure 4.6. The work flow of sender is also divided into three stages: prepare stage, authentication stage and sending stage.

Stage 1, like downloader, the first thing of prepare stage is to initiate process, such as create the server sslsocket and set some parameters. And then server will start to listen and wait to client (device) to connect with it. If there is any device requesting to connect with the server, then it turns to stage 2.

Stage 2, when server connects with client successfully, then the manager will authenticate with this device for security issue. The more detailed authentication process will be discussed in section 4.4.1. If authentication fails, then it will turn back to listening process of stage 1. Otherwise, it will proceed to stage 3.

Stage 3, the final sending function depends on two factors: the result of verification and authentication. Only when both of these two factors are true, the sending function can be executed. After file sent, the sender will refresh content of UI of Manager and display the information of newer version application.

4.4 Authentication

The authentication layer is very crucial to this system, because it is meaningless to a system without security guarantee. Figure 4.7 presents the overall security methods of this automatic update system. It can see that the security methods are divided into three categories: SHA, DSA and SSL. The theory of these methods has been discussed in section 2.2. Next three sections will describe the detailed implementation of these three methods.
4.4.1 Digital Signature

The main purpose of digital signature is used to verify the JAR file’s integrity in this system. After JAR file being generated, the JAR file needs to be signed. This can prevent device launch a corrupted JAR and occur unpredictable results. Figure 4.8 explains how this automatic update system implements digital signature and verification.

First, create a keystore using Keytool tool to generate the private key and public key (exported to certificate). Then sign the JAR file using Jarsigner tool with private key. Jarsigner is a signature tool based on Java KeyStore (JKS) which is repository of the security certificates. Jarsigner tool has two functions: To sign a JAR file and to verify the signed JAR file. The corresponding certificate is used to verify the signed JAR file.

**Signature**: This happens when Administrator chooses to upload JAR file, the program will sign the JAR using the private key specified by developer.
**Verification**: The certificate is sent to Manager and Device, which will be used to verify the signature. There are two cases to verify the JAR file. First, after the Manager downloading the JAR, it will perform the first verification. The reason for this is that the JAR file may be tampered by hiker in database or the JAR file was corrupted during downloading time. Second, after device receiving the JAR file, it will perform the second time verification. For that the JAR file may be corrupted during sending and receiving time.

### 4.4.2 Secure Sockets Layer
The ordinary socket only achieves the network communication without security guarantee. But the SSL socket can make up the limitation to implement secure network communication. The basic principle of SSL socket to reach the security is illustrated in figure 4.9 [24].

![Figure 4.9: The work flow of SSL for automatic update system.](image)

Before establishing connection, both of server (Manager) and client (Device) must have two files related to certificate. One file is own certificate, another file is the trusted list which records certificates it trusts. When the SSL server socket starts to listen and SSL client socket requests to connect, both of them will send its own certificate to other side. When they received the certificate, they will search all of trusted list to see if there exists the certificate of other side. If not, then just give up this connection and continue to next turn. If both of certificates
existed in the trusted list of other side, then server and client will establish a real connection.

4.4.3 Secure Hash Algorithm

The SHA authentication is triggered only when the SSL socket connects successful as figure 4.7. This is double security which can increase security level of this system largely. Figure 4.10 describes it in detail.

![Secure Hash Algorithm (SHA)](image)

**Figure 4.10: The work flow of SHA for automatic update system.**

The server (Manager) has a trusted device list (appears as a file) and maintains it by adding and removing device. Assuming every device has two properties: Name and Password. When server adds device, it will store the name and SHA1 value of the password to the trusted device list. When removes device, just delete the name and SHA1 value of password from trusted device list. The reason for this is that the file and the message from client may be compromised by hiker, so the device name and its corresponding password can be exposed. As a result, the hiker can perform malicious manner by posing the device. But if store and send the SHA value of password, these bad results can be avoided. After server and client establishing SSL socket connection, the client will send its name and SHA value of password to server. And server will compare it with key-value pair in its trusted device list after receiving the message from client. If there is the key-value pair, then the final result of authentication is success. Otherwise authentication fails.
4.5 Device

Device layer is the final layer of this automatic update system. From a functional perspective, Device has two main functions: Receive JAR files (Receiver) and installs it (Installer). Next two sections will follow figure 4.11 to describe how Device implement these two functions. The related source code is Device in Appendix A.

![Functional Flow Chart of Device]

4.5.1 Receiver

The purpose of receiver is to receive the JAR file from downloader of Manager. This part is illustrated in left side of figure 4.11. According to the main function of receiver, the process can be summarized as three stages. Prepare stage, receiving stage, and verification stage.

Stage 1, prepare stage covers the several sub processes. Firstly, like general process, it will initiate the needed parameters such as working directory. And launch the current application which is the newest version. It is worth noting that the application will always keep running after launching until installing newer application. Then the receiver looks for the specific server and request for connection. When
connection success, proceed to condition judgment “Authentication”, if pass it, go to the receiving stage, otherwise just go back.

Stage 2, in receiving stage, the receiver will receive the JAR file from downloader over TCP/IP network and store it in local place of device. This is critical step because the purpose of the previous work is to be able to successfully receive the lossless JAR file.

Stage 3, verification occurs after receiving the JAR file, the purpose of which is to ensure the integrity of JAR file. If the JAR file is damaged, then delete all of received files, restore the environment and go back to stage 1. If not, there are two things to do. Display some information about receiving file and triggers Installer.

4.5.2 Installer

The ultimate goal of this automatic update system is to install the new version application successfully. The process flow is shown in right side of figure 4.11. Installer starts after receiving an instruction from receiver. The installer includes also includes three stages: prepare stage, install stage and restart stage.

Stage 1, relatively speaking, prepare stage task of installer is easy. The installer just needs to initiate the process such as the JAR file source, location of log file and so on.

Stage 2, install stage is extremely important because the installation results are affected by many factors such as the running environment status, memory footprint and so on. This step that checks if there is running corresponding application before actual installation is necessary. If exists, the installer will kill it using appropriate ways. After above finished, installer executes “java –jar xxx.jar” command line in the background to install the JAR file. If success, jump to stage 3. Otherwise, jump out of installer process.

Stage 3, after installing the new version application, it is necessary reset running environment, so restart of device is necessary. After execute “Restart” command and device is powered on again, the device will launch the new version application.
5 Results

This chapter mainly shows the results as followed by chapter 4 except authentication layer which will be discussed in evaluation. These results mainly contain the User Interface and functions of Administrator, Manager and Device program and the database. In addition, some additional work such as log files will also be displayed.

5.1 Administrator

The Administrator presents the content of UI when uploading a new version application to database server. The Administrator is a tool which is designed to control application version updates. It can only be manipulated by the system administrator for secure issue. In order to facilitate the operation of the administrator, develop an “Administrator GUI” which is illustrated in figure 5.1. This GUI contains two parts: Display interface and Control interface.

The Display interface includes an icon at the top, and Application Name, Release Time, Current Version. In addition, it contains a note at the bottom used to remind the user.

![Figure 5.1: The result of the Administrator.](image-url)
The Control interface contains the “Upload” button, which is the key of the GUI as it can be used to achieve upload function. When administrator clicks this button, it will prompt to a “FileChooser” window.

As the figure 5.2, if the selected archive has existed, it will prompt to a warning window written “File existed! Please re-select”. If the selected archive is older than current, it will prompt to a warning box written “Upload version is older than current! Please re-select!”. If there is no problem with the uploaded archive, it will prompt to an upload success notification.

![Prompt box for different scenarios.](image)

**Figure 5.2: Prompt box for different scenarios.**

In order to facilitate the administrator to view the upload history, add the log function in the administrator program as shown in figure 5.3. This will record automatically the application’s source, release time, application name, version number and upload time every time when administrator uploads a file.

![Upload content in log_upload.txt.](image)

**Figure 5.3: Upload content in log_upload.txt.**
5.2 Database
The main function of Tomcat Server is to provide a database as shown in figure 5.4. The top of this figure is the website’s URL of the database. It is “http://localhost:9999/myAuto/Database/”. “localhost” is the host name and 9999 is the port number which can be configured. “myAuto” is the project name of server and “Database” is sub-folder of project. The middle of figure showed the content of database folder. This folder contains two types of files: JAR file and Properties file. There can exist multiple JAR files, which are the application files which can be executed, and these file are sorted by order. For Properties file, there is always only one this type of file, which can be considered as configuration file, because it always records the current newest version JAR file information. Properties file is extremely important for Manager to check the newest version JAR file.

![Table showing file names and sizes]

Figure 5.4: The application database in Tomcat server.

5.3 Manager
The result of Manager is shown in figure 5.5. The content of UI is divided into two categories following section 4.3: downloader and sender. The subsequent two sections will explain the detailed control interface and display interface according to figure 5.5.
5.3.1 Downloader

The button “Start” can trigger the downloader start to run. When click “Start”, a timer of downloading task will start and will re-run regularly. If manager wants to stop the timer only by pressing button “Stop”. Assuming the timer is active and there is newer version JAR file, then the downloader will download it to its local system and display “Download xxx_xxx successful!” .Then verify the integrity of file using related tool discussed in section 4.4.2. If it is successful, display “xxx_xxx verify successful” and refresh the “Information Display” which contains “App Name” and “Current” version number. As the same time, record the log file as figure 5.6. This log file presented the release time, application name and version number of the JAR file and the actual download time for downloader.

Figure 5.6: Download content in log_download.txt.
5.3.2 Sender

The sender starts automatically after the Manager program starting. It will print “Server is listening” and wait to device to connect. If there is any device to perform connecting, then these two sides will authenticate each other. If success, print “Authenticate success!” , otherwise “Authenticate fails!” . Only the result of authentication is success, the sender will wait for downloader to issue the sending command. If sender gets the command, then it will send file to device and report the sending result on UI.

The device management part on the bottom of UI is designed to manage the list of trusted device by “Add” and “Delete” device. “Delete” functions requires that the device name and password must be input right. And the “Add” function requires that the input of device name and password must not be null. The security implementation of add and delete devices was discussed in section 4.4.1.

5.4 Device

The result of Device is divided into two parts. One is the receiver’s results before installing. Another is the installer’s results when installing new application.

5.4.1 Receiver

The receiver process will start as long as the Device program starts up. As shown figure 5.7, there are three types of displays. The top of UI is designed to display the current running application information such as application name and current application version. The middle of UI is used to display the connection and download information. When Device connects with server successful, it will print “Connect server successful”. After connection, Device will authenticate with Manager each other, then it will print “Authenticate successful/failure!” according to different result.

As the same time, “Install information” text which is at the bottom of UI will present the information and total has been running time of current running application. Keep application running is normal in real life, so the running time is used to indicate that the application has been in progress rather than exit.

The receiving display is shown in figure 5.8. If this process goes on after authentication success, the receiver will always wait until the JAR file incomes. Then the receiver receives, stores the file and prints
“Download xxx successful!” Then verify it, if result appears to pass, then prepare to install.

![Image of device installation process](image)

Figure 5.7: The receiver’s result of the Device.

### 5.4.2 Installer

The installer is used to install new application. The install result is shown in “Install Information” text in figure 5.8. When device begins to install application, the “Install Information” will indicates “Prepare to install xxx_xx”. And print the application information such as name, vendor, location, license and so on. Then the receiver will load the JAR file and starts the real installation. In order to better observe the installation process, it will output the installation progress.
Figure 5.8: The installer’s result of the Device.

When installation finished, the process outputs “Install End”. And log the install information of specific application as shown in figure 5.9. This file will record the application name, version and installation time. As the same time, issue the “Restart” command to process. When process executes the “Restart” instruction, all of content on UI will be cleared and the Device will be restarted. When restarts over, the Device will launch the newest version application.

Figure 5.9: The content of log_install.txt.
6 Evaluation

This chapter contains measurement and evaluation for performance and security. The purpose of this section is to evaluate the performance and security level for this system using some technical and mathematical methods. The scope of the performance assessment is gathered on the latency. And the assessment of security focuses on the results of signature, verification and authentication.

6.1 Performance

For an automated update system, the timeliness of system updates is often important. If the developer released a new version due to the serious bug of old version, but the huge delay caused the device did not install the new version successfully in a certain period of time, which may cause serious consequences. So before the system is officially released, the testing of various delays is very necessary, which will ultimately determine whether an automatic update system can be put into real life.

6.1.1 Method

There are a variety of delays in this project, but to calculate all the delay is unrealistic. So select several delays which perform a greater impact on the system performance. As shown in the following table 6.1, download, manager verification, transmission, device verification, installation, restart and actual total latency is listed to measure. The type 1, 2, 3, 4, 5, 6 are the important intermediate delay in the automatic update system. Latency type 7 is the actual total delay from administrator uploading JAR successfully to device restarts over. Actual total delay is the most important delay and relating to performance of system directly. There are three goals needed to achieve for evaluation of performance:

Goal 1: Find the intermediate delay that is most relevant to the actual delay.

Goal 2: Find the most related cause for the fluctuation of actual total latency.

Goal 3: Assess the performance of this automatic update system according measurement result.
Table 6.1 Measurement method of each type of latency

<table>
<thead>
<tr>
<th>Num.</th>
<th>Start time</th>
<th>End time</th>
<th>Latency types</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Administrator upload successfully</td>
<td>Manager download successful</td>
<td>Download</td>
</tr>
<tr>
<td>2</td>
<td>Manager verify start</td>
<td>Manager verify end</td>
<td>Manager verification</td>
</tr>
<tr>
<td>3</td>
<td>Manager sends JAR successfully</td>
<td>Device receive JAR successfully</td>
<td>Transmission</td>
</tr>
<tr>
<td>4</td>
<td>Device verify start</td>
<td>Device verify end</td>
<td>Device verification</td>
</tr>
<tr>
<td>5</td>
<td>Device installation starts</td>
<td>Device installation ends</td>
<td>Installation</td>
</tr>
<tr>
<td>6</td>
<td>Device restart starts</td>
<td>Device restart over</td>
<td>Restart</td>
</tr>
<tr>
<td>7</td>
<td>Administrator upload successfully</td>
<td>Device restart over</td>
<td>Actual total</td>
</tr>
</tbody>
</table>

6.1.2 Measurement

It is worth noting that the check timer of Manager is set to 10 seconds. So the Manager will always check database every ten seconds. This timer is important for measurement result. Measure these types of latency according to method provided by table 6.1, in addition, add a row named total which is the total number delay for six intermediate delays. And then compute their average delay and standard deviation for each type of latency, the results is shown in table 6.2.

It is easy to note that the transmission delay is 0, and manager verification and restart delay don’t have any changed all the time. So the standard deviation of these three types of latency is 0, which is staggering stability. For device verification and installation latency, the standard deviation varies a little, they keep relatively stable. However, for the download, total and actual total delay, the standard deviation of them varies a lot, and appears very unstable. So the reliability can be divided into three categories according to standard deviation: Stable, relatively stable, unstable.
Table 6.2 Measurement according to latency type

<table>
<thead>
<tr>
<th>Latency/s</th>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
<th>4th</th>
<th>5th</th>
<th>6th</th>
<th>Average</th>
<th>Stdev</th>
</tr>
</thead>
<tbody>
<tr>
<td>Download</td>
<td>7</td>
<td>3</td>
<td>8</td>
<td>1</td>
<td>1</td>
<td>7</td>
<td>4.5</td>
<td>3.209361</td>
</tr>
<tr>
<td>Manager verification</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Transmission</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Device verification</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1.66667</td>
<td>0.516398</td>
</tr>
<tr>
<td>Installation</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4.16667</td>
<td>0.408248</td>
</tr>
<tr>
<td>Restart</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>Total</td>
<td>20</td>
<td>16</td>
<td>21</td>
<td>14</td>
<td>14</td>
<td>19</td>
<td>17.3333</td>
<td>3.076795</td>
</tr>
<tr>
<td>Actual Total</td>
<td>21</td>
<td>17</td>
<td>22</td>
<td>14</td>
<td>14</td>
<td>20</td>
<td>18</td>
<td>3.521363</td>
</tr>
</tbody>
</table>

6.1.3 Analysis

In order to better observe the measurement results, visualize the results of Table 6.2 to line chart as figure 6.1. It is easy to observe that the trend of download, total and actual delay is same from this visualization result. So there is reason to believe that there is a huge link between these three delays. What is more, the total delay is just arithmetic results from six intermediate delays, so only the download delay can be considered as the most relevant intermediate delay. And basically it can be said that there is no link between other delays.
Figure 6.1: The visualization of measurement results.

For the goal 1, the answer can be found from the visualization result, it is the download delay.

For the goal 2, it can draw this conclusion that the download delay causes the huge fluctuation of actual total delay. This is because the check timer of Manager is 10 seconds. So when JAR file uploaded, the timer may be at sleep, or is about to start. This will cause the actual time vary a lot.

For the goal 3, the average actual total delay is 18 seconds, it is a good result based on 10 second’s timer. Finally, this system can be seen as a low-delay system, to some extent, the performance is good.

Figure 6.2: The boxplot of most related delays for performance.
As can be seen from Figure 6.1, the intermediate delay most associated with the total delay is the download delay. In order to better show the delay of the fluctuations, make the boxplot Figure 6.2 of download, total and actual total.

### 6.2 Security

For a system, it is meaningless to fully realize the expected functionality but does not guarantee security. Therefore, the purpose of this project in addition to the automatic update function, security is also an important factor in one of the considerations. That is, the main function of this automatic update system is divided into two parts: security and automatic updates. This section evaluates the security of the system to see if it meets the intended goal.

#### 6.2.1 Methods

This automatic update system applies three technologies to ensure security: DSA, SSA, and SHA. In this system, each technology has not exactly the same focus which is listed in table 6.3. The evaluation focuses on the results of verification, connection and authentication. In verification stage, this system achieved the integrity and non-repudiation to ensure the installed JAR is not corrupted and source of JAR is right. In connection stage, this system guaranteed the authenticity and confidentiality to ensure that the other side is own trusted object and all transmitted content will be encrypted by session keys. In authentication stage, the system ensured the authenticity through the device trusted list of Manager.

<table>
<thead>
<tr>
<th>Object</th>
<th>Technology</th>
<th>Stage</th>
<th>Guarantee</th>
</tr>
</thead>
<tbody>
<tr>
<td>JAR</td>
<td>Digital signature algorithm</td>
<td>Verification</td>
<td>Integrity, Non-repudiation</td>
</tr>
<tr>
<td>Manager &amp; Device</td>
<td>Secure sockets layer</td>
<td>Connection</td>
<td>Authenticity, Confidentiality</td>
</tr>
<tr>
<td>Manager &amp; Device</td>
<td>Secure hash algorithm</td>
<td>Authentication</td>
<td>Authenticity</td>
</tr>
</tbody>
</table>

#### 6.2.2 Measurement

Measure result of each stage with different pre-condition, and compute their total success rate as table 6.4 list. For the verification stage, did five times of corrupted JAR and ten times of corrupted JAR to computer the
success rate. For connection stage, change the pre-condition to right Manager && Device and wrong Manager or Device. For authentication stage, it is same as connection stage.

<table>
<thead>
<tr>
<th>Stage</th>
<th>Object</th>
<th>Frequency</th>
<th>Success rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verification</td>
<td>Uncorrupted JAR</td>
<td>5</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>Corrupted JAR</td>
<td>10</td>
<td>0%</td>
</tr>
<tr>
<td>Connection</td>
<td>Right Manager &amp; Device</td>
<td>5</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>Wrong Manager or Device</td>
<td>10</td>
<td>0%</td>
</tr>
<tr>
<td>Authentication</td>
<td>Right Manager &amp; Device</td>
<td>5</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>Wrong Manager or Device</td>
<td>10</td>
<td>0%</td>
</tr>
</tbody>
</table>

### 6.2.3 Analysis

The success rate of right pre-condition is 100%, and is 0% in other cases, which is a gratifying result. To a certain extent, the system can be considered enough secure, because it achieved the integrity, non-repudiation, authenticity and confidentiality. However the system may be compromised in for some extreme conditions, such as certificate forgery, because the system cannot identify this fake certificate which appears with right one.
7 Conclusions

The overall aim of this project is to evaluate and implement a secure automatic update system based on the Internet of things. The concrete goals of this project (see Section 1.3) are verified as following:

Goal 1, Find three automatic update existing works was reached. These three related works were discussed as section 2.4, and they were “Windows Update”, “Java Web Start” and “Getdown” respectively.

Goal 2, Find finding three solutions for ensuring security was also reached. These three secure methods were explained in section 2.2, they were secure hash algorithm, digital signature and secure sockets layer. And the evaluation for security has demonstrated their enough security to protect a system.

Goal 3, this also was achieved, this report proposed an overall network architecture in chapter 4 according to the combination of three existing related works and three secure methods. The manager and device always connect and authenticates with each other, as the same time, the manager will check the database every in a period. When the administrator uploads an application to the database, the manager will download and send it to device. If goes smoothly, the device will install the application and restart itself in the end.

Goal 4, the scenario is assumed in industrial internet of things. The system has been implemented using the Java programming language and can successfully perform secure automatic updates to the Java program, The implementation approaches were explained technically in chapter 4 in detail.

Goal 5, the performance and security had been evaluated. The merits of performance are the actual total latency and intermediate latency. The measured results indicated that the automatic update system can be finished within twenty seconds, which is a short period, so the system can be considered as good performance. As for the security, measured the automatic update system success rate of three different stages of different objects, the results proves that the automatic update system is enough secure.
7.1 Ethical discussion
The rapid development of the Internet of Things brings benefits to mankind, at the same time also led to a series of social ethics discussion. For example, once the Internet of Things becomes ubiquitous, human privacy will be violated heavily. Automatically update for industrial Internet of things is the same as Internet of things, will bring ethical issues. It has three main following ethical issues:

First, increase the unemployment rate, because the automation update will replace the manual update, which means the replacement of this post staff.

Second, the system stability is reduced, once the system vulnerabilities are used, the scope of the damage will be all devices of the factory, and even leads to the entire factory paralysis.

Third, it is about the responsibility issue. Who should be accountable for bad update result made by this automatic update system? It is the developer for this automatic update system or the factory as using the easy compromised operating system?

7.2 Future work
Although the system can perform both of automatic update and security, there are still lots of work can be done in the future.

For the update file, this system is for a single JAR file, but most of the installation files are multiple and different forms. So if there is more time, this system could be extended from single JAR file to multiple related files.

For performance of this system, the availability and the scalability is difficult to be evaluated. Indeed, these two merits were also important for industrial system. For the availability, maybe the backup service can be considered in the future. As for the scalability, more simulated devices should be developed and try to connect with this system simultaneously.

For the security, the drawback of this system is that it cannot recognized the malicious manner if the hiker owns the legal certificate. If this system is attacked by this type of means, the attacker can steal the data without being found. So this type of attack should be considered and increase the corresponding security policy.
References


Appendix A: Source Code

https://github.com/MollyMin/Auto